High Level Design & Low Level Design

The purpose of this document is to provide a template for documenting both HLD & LLD.

**Document Control :**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | | | |
|  | | |  | |  | |  |  |  | |  | |  | |
|  | | |  | |  | |  |  |  | |  | |  | |
|  | | |  | |  | |  |  |  | |  | |  | |
| **Date** | | | **Version** | | **Author** | | **Brief Description of Changes** | | | | **Approver Signature** | | | |
| 28/11/2022 | | | 0.1 | | Whole Team | | SRS Briefing | | | |  | | | |
|  | | |  | |  | |  | | | |  | | | |
|  | | |  | |  | |  | | | |  | | | |

[1. Introduction 3](#_Toc368912248)

[1.1. Intended Audience 3](#_Toc368912249)

[1.2. Acronyms/Abbreviations 3](#_Toc368912250)

[1.3. Project Purpose 3](#_Toc368912251)

[1.4. Key Project Objectives 3](#_Toc368912252)

[1.5. Project Scope and Limitation 3](#_Toc368912253)

[1.5.1. In Scope 3](#_Toc368912254)

[1.5.2. Out of scope 3](#_Toc368912255)

[1.6. Functional Overview 3](#_Toc368912256)

[1.7. Assumptions, Dependencies & Constraints 3](#_Toc368912257)

[1.8. Risks 3](#_Toc368912258)

[2. Design Overview 3](#_Toc368912259)

[2.1. Design Objectives 3](#_Toc368912260)

[2.1.1. Recommended Architecture 3](#_Toc368912261)

[2.2. Architectural Strategies 3](#_Toc368912262)

[2.2.1. Design Alternative 3](#_Toc368912263)

[2.2.2. Reuse of Existing Common Services/Utilities 3](#_Toc368912264)

[2.2.3. Creation of New Common Services/Utilities 3](#_Toc368912265)

[2.2.4. User Interface Paradigms 3](#_Toc368912266)

[2.2.5. System Interface Paradigms 3](#_Toc368912267)

[2.2.6. Error Detection / Exceptional Handling 3](#_Toc368912268)

[2.2.7. Memory Management 3](#_Toc368912269)

[2.2.8. Performance 3](#_Toc368912270)

[2.2.9. Security 3](#_Toc368912271)

[2.2.10. Concurrency and Synchronization 3](#_Toc368912272)

[2.2.11. Housekeeping and Maintenance 3](#_Toc368912273)

[3. System Architecture 3](#_Toc368912274)

[3.1. System Architecture Diagram. (Not Necessary) 3](#_Toc368912275)

[3.2. System Use-Cases 3](#_Toc368912276)

[3.3. Subsystem Architecture 3](#_Toc368912277)

[3.4. System Interfaces 3](#_Toc368912278)

[3.4.1. Internal Interfaces 3](#_Toc368912279)

[3.4.2. External Interfaces 3](#_Toc368912280)

[4. Detailed System Design 3](#_Toc368912281)

[4.1. Key Entities 3](#_Toc368912282)

[4.2. Detailed-Level Database Design 3](#_Toc368912283)

[4.2.1. Data Mapping Information 3](#_Toc368912284)

[4.2.2. Data Conversion 3](#_Toc368912285)

[4.3. Archival and retention requirements 3](#_Toc368912286)

[4.4. Disaster and Failure Recovery 3](#_Toc368912287)

[4.5. Business Process workflow 3](#_Toc368912288)

[4.6. Business Process Modeling and Management (as applicable) 3](#_Toc368912289)

[4.7. Business Logic 3](#_Toc368912290)

[4.8. Variables 3](#_Toc368912291)

[4.9. Activity / Class Diagrams (as applicable) 3](#_Toc368912292)

[4.10. Data Migration 3](#_Toc368912293)

[4.10.1. Architectural Representation 3](#_Toc368912294)

[4.10.2. Architectural Goals and Constraints 3](#_Toc368912295)

[4.10.3. Logical View 3](#_Toc368912296)

[4.10.4. Architecturally Significant Design Packages 3](#_Toc368912297)

[4.10.5. Data model 3](#_Toc368912298)

[4.10.6. Deployment View 3](#_Toc368912299)

[5. Environment Description 3](#_Toc368912300)

[5.1. Time Zone Support 3](#_Toc368912301)

[5.2. Language Support 3](#_Toc368912302)

[5.3. User Desktop Requirements 3](#_Toc368912303)

[5.4. Server-Side Requirements 3](#_Toc368912304)

[5.4.1. Deployment Considerations 3](#_Toc368912305)

[5.4.2. Application Server Disk Space 3](#_Toc368912306)

[5.4.3. Database Server Disk Space 3](#_Toc368912307)

[5.4.4. Integration Requirements 3](#_Toc368912308)

[5.4.5. Jobs 3](#_Toc368912309)

[5.4.6. Network 3](#_Toc368912310)

[5.4.7. Others 3](#_Toc368912311)

[5.5. Configuration 3](#_Toc368912312)

[5.5.1. Operating System 3](#_Toc368912313)

[5.5.2. Database 3](#_Toc368912314)

[5.5.3. Network 3](#_Toc368912315)

[5.5.4. Desktop 3](#_Toc368912316)

[6. References 3](#_Toc368912317)

[7. Appendix 3](#_Toc368912318)

# 

# Introduction

This messaging application project is used to communicate between two or more users by the messaging application. In this application users can communicate and can share files and those will be stored on the server so that we can share those files through the server.

## 1.1. Intended Audience

This Application is intended to be read by the client.

|  |  |
| --- | --- |
| Val grind |  |
| CPP |  |

## 1.2. Acronyms/Abbreviations

|  |  |
| --- | --- |
| UT | Unit Test |
| IT | Integrated Test |
|  |  |
|  |  |

## 1.3. Project Purpose

The purpose of Custom Messaging Application is, it makes easy to communicate with people anywhere in the world by sending and receiving messages in real time. With this chat Application, users can receive the same engaging and lively interactions through custom messaging features, just as they would in person.

## 1.4. Key Project Objectives

Users can create their own id and Login to the Application.

Server will Provide the Active user list to the client.

Client Choose whom they want to chat from the Active user list.

They can Exchange messages and share files with another users.

## 1.5. Project Scope and Limitation

Custom Messaging Application simply provides the service for users to chat with other users and get connected to them and share files with them. But they must connect with a server to get exchange messages.

### 1.5.1. In Scope

The scope of the Custom Messaging Application makes it easy to communicate with people anywhere in the world by sending and receiving messages. The messages will be exchanged via client server socket mechanism. This will enable the two systems to talk to each other in real time.

### 1.5.2. Out of scope

Required to involve techniques such as socket programming in TCP, process management to make the project done completely.

## 1.6. Functional Overview

## CPP ATL enables to code the job specifications, File IO operations help to read file from Command line arguments and write schedule into different text files. Val grind captures the data of memory leak.

## 1.7. Assumptions, Dependencies & Constraints

Assuming we are sending files from one user to another user and sending message.

It depends on the server to send or receive messages.

## 1.8. Risks

All assumptions, functional overview and design parameters are documented without evaluation which are to be implemented without missing.

# 2. Design Overview

# The Custom Messaging Application should display a main menu to User that shows the option to Create User id and password if the user is logging for the first time. After creating user id and password user data will be stored and monitored based on login and logout.

The server will match the username entered with the user list and if found, it will send a message to client that username already exists.

After Successfully login user will display the active user list and given the option to select the chat option, whether its private chat or group chat. The user can select any person from the list and continue to talk with him.

User can log off from a private chat by giving some command and the other users will be notified and exit from private chat and his data will be removed from the server.

Users will be displayed the Active group names and can select a group or create a new group. If another client selected the same group, then he will join, and this notification will be sent to all people in the group.

Users can start messaging and exit by giving a command. Users' data will be removed from the server once he types exit command.

Users can send files to the server and store them in the server in a separate folder by giving them a complete path name and then they can start sharing the files. If the file name is incorrect in sending the data, then user will exit from app.

## 2.1. Design Objectives

This Application Provides Real time messaging services where users can connect them with other users they want to connect and chat with them and share files with them.

### 2.1.1. Recommended Architecture

### UML Architecture.

UML stands for Unified Modelling Language. Any real-world system is used by different users. The users can be developers, testers, businesspeople, analysts, and many more. Hence, before designing a system, the architecture is made with different perspectives in mind. The most important part is to visualize the system from the perspective of different viewers. The better we understand the better we can build the system.

## **2.2. Architectural Strategy**

## Using UML Diagram, we can code for this project using C++ programming Language.

### 2.2.1. Design Alternative

Designed sequence diagram and use case diagram as design paradigm but as an alternative selected class diagram to visualize more data that have used in application.

### 2.2.2. Reuse of Existing Common Services/Utilities

Design and development are done from scratch using existing sources star UML for design and VI for development.

### 2.2.3. Creation of New Common Services/Utilities

Used existing resources to develop the application specific services

### 2.2.4. User Interface Paradigms

Client: Client are key part of the system. They should be able to choose the user to

whom they want to talk to.

Server: Server has access to client data. Server is responsible for managing and storing the client data.

### 2.2.5. System Interface Paradigms

### The system Interface paradigm implemented is a Menu driven interface between user and client. It displays a set of menus, which is to be chosen by the user and implementing application accordingly.

### 2.2.6. Error Detection / Exceptional Handling

Val grind application used to check for memory leaks if any and rectified if any memory leaks occur. Val grind is a programming tool for memory debugging, memory leak detection, and profiling. Val grind was originally designed to be a free memory debugging tool for Linux

### 2.2.7. Memory Management

GHz processor, 4GB RAM or more (system memory).

20 GB Hard Disk Drive or more.

1024×768 screen resolution.

Internet Connectivity.

PC with some equipment.

.

**2.2.8. Performance**

The performance of this application is quite simple. When a user logs in to an application it gets connected to the server and after that user can choose any person or group whom they want to chat and share files with them.

### 2.2.9. Security

When client login in application and when they send some files to other user then the files first stored in server with a separate folder with some name and when user enter the wrong folder name they got exited from the application and their data will be removed from the server.

**2.2.10. Concurrency and Synchronization**

Application is designed to work concurrently without interfering with the function of other part of code and simultaneously.

**2.2.11. Housekeeping and Maintenance**

Memory: System will have only 10GB space for data server.

Language Requirement: software must be in only English.

Implementation constraints: Application should be based on CPP.

# 3. System Architecture

UML design is the shortest form of “Unified Modelling Language.” The purpose of this modelling language is to visualize the design of the system. There are 14 types of UML diagram. They are:

* Use case diagram
* Sequence diagram

Class diagram

# 

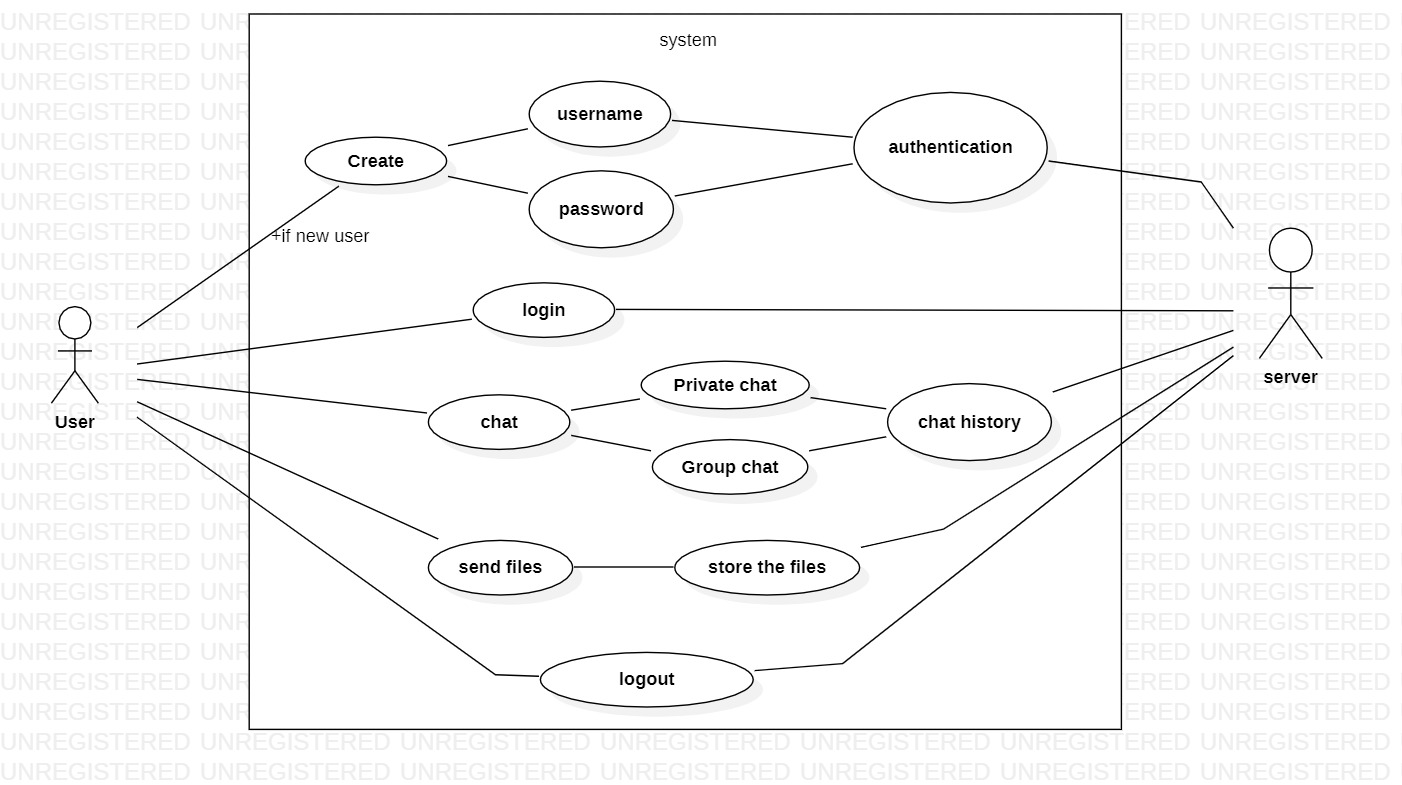
# 3.1. System Architecture Diagram. (Not Necessary)

## 3.2. System Use-Cases

## 3.2.1. Use case diagram

The use case diagram is one of the methods used to show the processes involved in the system. It depicts the system's structure and behavior. Additionally, the diagram consists of processes (use case) and users, or actors. It uses defined symbols to describe the overall flow of the system.

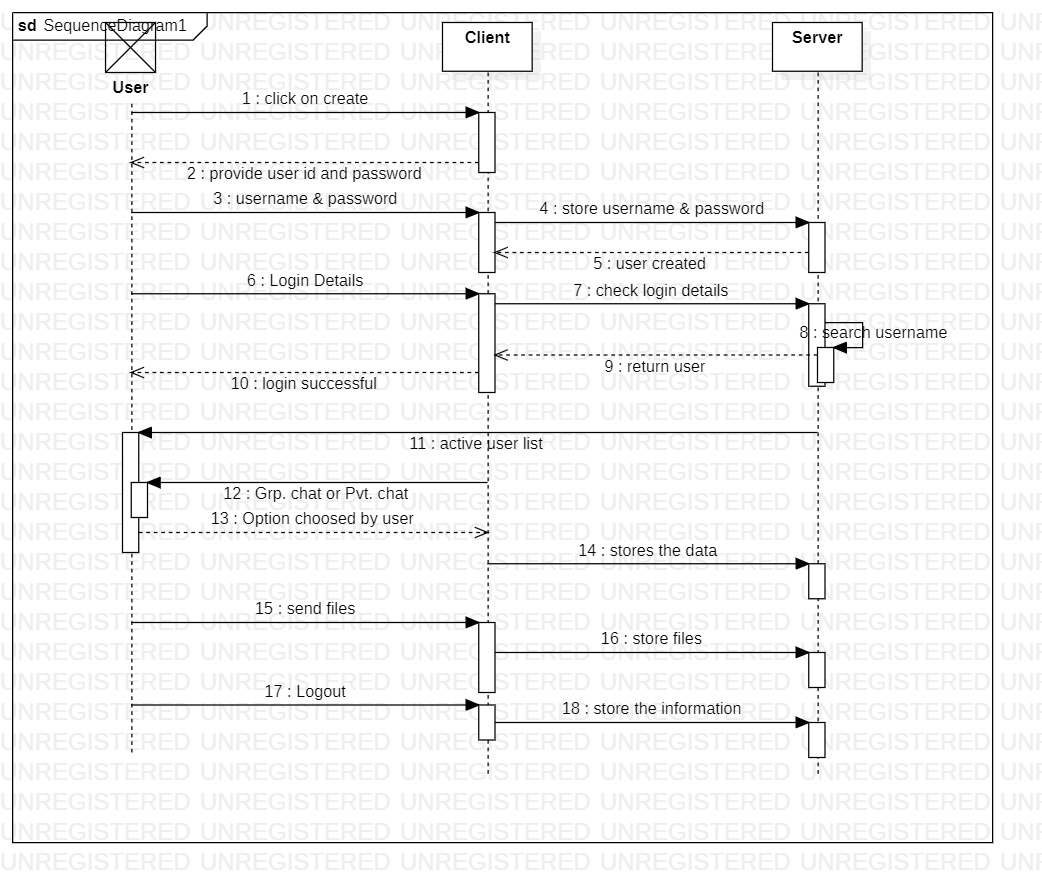
Here we are taken Actor as user and server. In between user and server all the functions will be done. If the username entered does not exist in the server, it will be created. All the login and log out information will be stored on the server. Users can chat with the other users by selecting the chat option, it may be private chat or group chat. And can share the files and those files will be stored in the server.



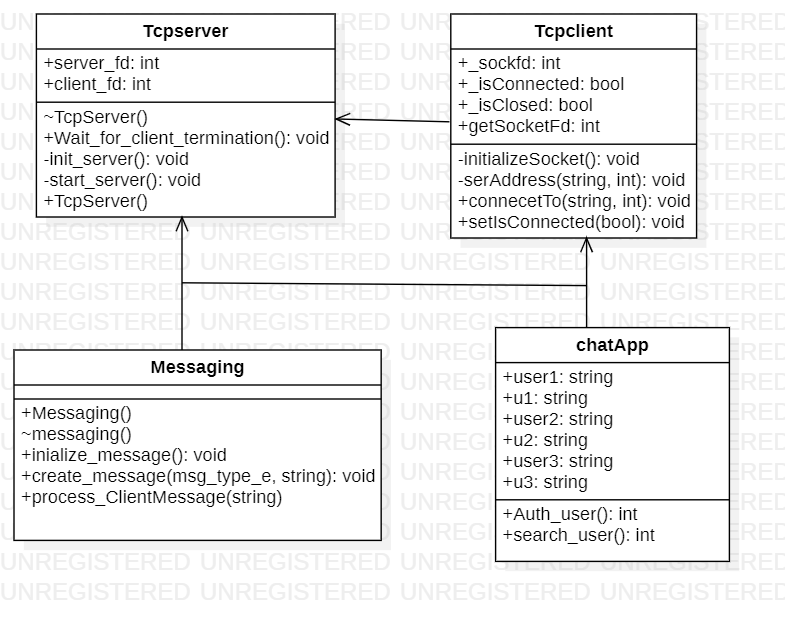
**3.2.2. Sequence diagram**

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects represented by lifelines and the messages they exchange over time during the interaction.

A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.



**3.2.3. Class diagram**

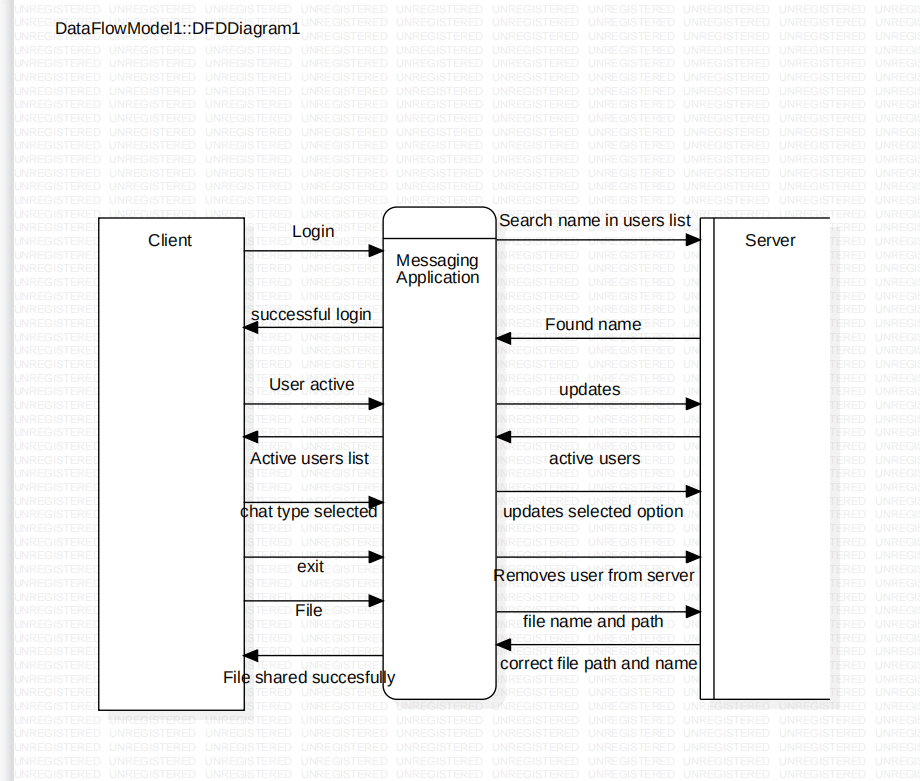


**3.3. subsystem-Architecture**

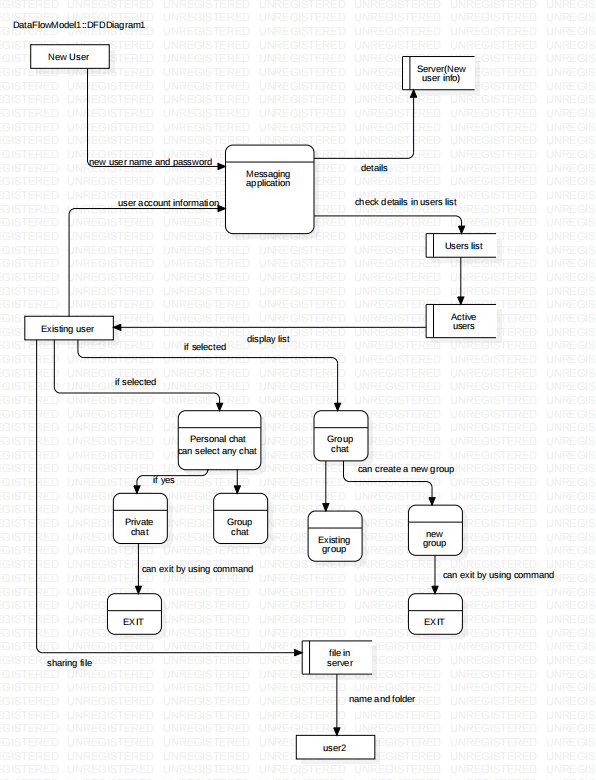
**3.3.1. Data flow diagram**

Data flow diagram (DFD) is a visual aid to see the information flow within a system. It shows how data enters and leaves the system, what changes the information and flow of information, and where data is stored.

**3.3.1.1. Level 0 Diagram**



**3.3.1.2. Level 1 Diagram**



**3.4. System interfaces**

### · Supportability: The application is easy to use.

### · Design Constraints: The system is built using CPP and system programming language.

### · Reliability & Availability: The application is active 24/7, that is whenever the user would like to chat with someone, they can use it up to its functionalities.

### · Performance: The system will work on the user’s and server/client.

**3.4.1. Internal interfaces**

P thread

CPP File handling

System Programming

CPP Language

**3.4.2. External interfaces**

Hardware Interfaces

* GHz processor, 4 GB RAM or more (system memory)
* 20 GB of hard-drive space or more
* VGA capable of 1024×768 screen resolution
* Necessary computer peripherals such as keyboards etc.
* Internet Connectivity (Wired/ Wireless)

Software Interfaces

* Windows/ Linux Based OS/ Mac OS/ Any OS capable of running c ++
* Database
* Server

# 4. Detailed System Design

Here, detailed information of low-level designs is described. Low-level designs include attribute, particulars that are used while implementing application

## 4.1. Key Entities

## Register, Login, Logout, Files, user id, password.

## 4.2. Detailed-Level Database Design

Here, we use files to store the data of registered users.

## 4.3. Archival and retention requirement

Not applicable for this project

## 4.4. Disaster and Failure Recovery

Not applicable for this project

## 4.5. Business Process workflow

The workflow of our application mainly involves sockets in between client and server. Here client is provided with menu-based interface to select his/her requirement according to given menu.

The data which is given by the client includes the type of message, length and value of the message to be sent to the server. The data given by the user is stored in maps STL and sent to the server. Based on the selection of clients the server responds with suitable messages.

## 4.6. Business Process Modeling and Management (as applicable)

The business model used is Agile methodology, where the application is iterated after implementing each process in between the actual implementation of application. The steps implemented are the planning phase, which involves the detailed understanding of SRS and further documenting the system requirement specification. The next step is analyzing requirements and resources and then designing phase where UML diagrams are used. The next step is actual application implementation. The application is iterated for each phase of implementation.

## 4.7. Business Logic

In this section, we are going to represent the important entities implemented in business logic while implementing application.

## 4.8. Variables

## 4.9. Activity / Class Diagrams (as applicable)

## 4.10. Data Migration

Not applicable for the project.

### 4.10.1. Architectural Representation

Architecture representation includes use-case, sequence, dataflow diagrams of different levels. These are designed and elaborated in the respective sections accordingly.

### 4.10.2. Architectural Goals and Constraints

The main goal of designing different UML diagrams is to understand the key entities that are to be implemented in the application and understand the workflow while implementing the application. They help us to understand the step-by-step process to be followed by the developer. The constraints are, sometimes the implemented UML doesn’t represent the system functionality perfectly.

### 4.10.3. Logical View

### 4.10.4. Architecturally Significant Design Packages

Not applicable to this project

### 4.10.5. Data model

Not applicable to this project

**Legacy system data model**

**Proposed system data model**

**Interface data model**

### 4.10.6. Deployment View

Not applicable to this project

# 5. Environment Description

This section involves the hardware environment used and required to implement our application.

## 5.1. Time Zone Support

Indian Standard Time

## 5.2. Language Support

## CPP on Linux, System Programming, IPC.

## 5.3. User Desktop Requirements

## We use cloud machine and WinSCP as desktop requirements.

## 5.4. Server-Side Requirements

The data files require around 5MB of memory.

### 5.4.1. Deployment Considerations

Hardware, Software, File Storage, Session Storage

### 5.4.2. Application Server Disk Space

Not applicable to this project.

### 5.4.3. Database Server Disk Space

Not applicable to this project.

### 5.4.4. Integration Requirements

Integration is connecting systems, applications, and devices together so that you have a better flow of data and processes.

### 5.4.5. Jobs

Not applicable for this project.

### 5.4.6. Network

Network used to implement this application is TCP LAN

### 5.4.7. Others

Not Applicable

## 5.5. Configuration

Operating system, Processor.

### 5.5.1. Operating System

Linux

### 5.5.2. Database

Operating system, Processor, memory.

### 5.5.3. Network

It is a technique used for planning and scheduling large projects in such a way that it minimizes trouble spots such as delays and interruptions by determining the critical factors and coordinating. various parts of overall job.

Network is a process of assigning network settings, policies, flows and controls. In a virtual network, it is easier to make network configuration changes because physical network devices appliances are replaced by software removing the need for extensive manual configuration.

### 5.5.4. Desktop

LINUX, OS.

# 6. References

# 7. Appendix

**Change Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QMS Template Version Control (Maintained by QA)** | | | | | |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
| 28/11/2022 | 0.1 | Whole team | | Initial Version | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |